COBB Tips & techniques for MS-DOS and PC-DOS

May 1990
Vol. 1 No. 1

Inside DOS™

Finding missing files with the CHKDSK and FIND commands

The idea for this article was submitted by contributing editor Tim Landgrave

If you are like most people, you occasionally forget where you've stored a particular file on your hard disk. You might remember the name (or a portion of the name) of the file you're looking for, but you can't recall in which directory the file exists. In situations like these, you can waste a lot of time searching through your disk's directories until you find the file you're looking for.

Fortunately, DOS offers a couple of tools that make it easy to locate a particular file on your disk as long as you know some portion of the file name. In this article, we'll introduce you to these tools and show you how to use them to locate lost files.

Figure A

```
          C:\TD\MSW5DEVD.FIL
          C:\TD\VENTURA.FDF
          C:\TD\MSWORD5.EXM
          C:\TD\WINDOW2.EXM
          C:\TD\WINDINST.EXM
          C:\TD\WINSTUB.EXE
          C:\TD\WNMGR.EXM
          C:\TD\TYPEDIR.EXE
          C:\TD\MSW5PCSD.FIL
          C:\TD\MSW5RESI.FIL
          C:\TD\TD.ENV
          C:\CONFIG.SYS
          C:\COMMAND.COM
          C:\AUTOEXEC.BAT

 33421312 bytes total disk space
  3876864 bytes in 4 hidden files
    81920 bytes in 34 directories
 19666944 bytes in 922 user files
  9795584 bytes available on disk

   655360 bytes total memory
   572368 bytes free

C>_
```

You can display the full path name of every file on a disk by appending the option /v to the CHKDSK command.

The CHKDSK command

One of DOS's most useful commands is CHKDSK. You have probably used the CHKDSK command in the past to see how much free disk space or free memory is available on your machine. A seldom used feature of the CHKDSK command, however, is its /v option. When you append the /v option to the end of a CHKDSK command, in this way

```
   C>chkdsk c: /v
```

DOS will list the full path name of every file stored on drive C. After listing all the file names, DOS will display the same information that you normally see when you issue the CHKDSK command without the /v option.

For example, Figure A shows the text we saw after entering the command *chkdsk c: /v*. Unless your disk contains very few files, several screens of file names will scroll by faster than you can read them. If you want to read all the file names returned by the CHKDSK command, you must print the list (see the technique explained in the article "Redirecting DOS Output from the Screen to the Printer," on page 7.)

By the way, CHKDSK is an external DOS command—DOS cannot execute it unless it can find the file CHKDSK.COM. For this reason, make sure that your computer's path includes the directory that contains your DOS files, as explained in the article "Using the Path Command to Access External Commands," on page 6.

Continued on page 2

IN THIS ISSUE

Inside DOS

Inside DOS (ISSN 1049-5320) is published monthly by The Cobb Group, Inc., 9420 Bunsen Parkway, Suite 300, Louisville, KY 40220. Domestic subscriptions: 12 issues, $39. Foreign subscriptions: 12 issues, $59. Single issues: Domestic, $4 per copy; Foreign, $6.50 per copy. Send subscriptions, fulfillment questions and requests for bulk orders to Customer Relations, 9420 Bunsen Parkway, Suite 300, Louisville, KY 40220, or call toll free 1-800-223-8720. Address correspondence and requests for special permission to The Editor, *Inside DOS*, to the address above. **Postmaster:** Send all address changes to *Inside DOS*, P.O. Box 35160, Louisville, KY 40232. Second Class postage is pending at Louisville, KY.

Editor-in-Chief:	Mark Crane
Contributing Editors:	Van Wolverton
	Chris DeVoney
	Tim Landgrave
Editing:	Jim Welp
	Martha A. Clayton
Production:	Elayne Noltemeyer
Design:	Karl Feige
Publisher:	Douglas Cobb

Conventions

To avoid confusion, we would like to explain a few of the conventions we use in *Inside DOS*.

As you probably know, Microsoft has released several versions of DOS. Although, most of the articles in this journal apply to all versions of DOS since 2.00, we'll sometimes point out that a particular command or technique requires a particular version. For simplicity, we'll refer to all 4.xx versions as Version 4.

Typically, we'll use DOS's default prompt, C>, to represent the DOS prompt in our examples. If you have defined a custom DOS prompt, just assume that the C> represents your prompt.

When we instruct you to type something, those characters will usually appear on a separate line along with the DOS prompt. The characters you type will appear in color, while the characters DOS displays will appear in black.

Occasionally, we won't display the command you type on a separate line. In these cases, we'll display the characters you type in italics. For example, we might say, "Issue the command *dir *.txt* at the DOS prompt." Although DOS is not sensitive to capitalization, we'll always display the characters you type in lowercase.

When we refer to a general DOS command (not the command you actually type at the DOS prompt), we'll display that command name in all caps. For example, we might say, "You can use either the COPY or XCOPY command to transfer files from one disk to another."

Many commands accept parameters that specify a particular file, disk drive, or other option. When we show you the form of such a command, its parameters will appear in italics. For example, the form of the COPY command is

```
    copy file1 file2
```

where *file1* and *file2* represent the names of the source file and the target file, respectively.

The names of keys like [Shift], [Ctrl], and [F1] appear are enclosed in brackets. The [Enter] key is represented by the symbol ↵. When two keys must be pressed at the same time, those key names will appear side by side, as in [Ctrl][Break] or [Ctrl]z.

CHKDSK and FIND commands

Continued from page 1

Finding a particular file

As we hinted earlier, you can use the result of the CHKDSK command along with another helpful tool—the FIND command—to locate a particular file on your disk. To do this, you must use a command that takes the form

```
    chkdsk c: /v | find "string"
```

where *string* is a known string of Uppercase characters that appears in the file's name. Notice that this command "pipes" the output of the CHKDSK command to the FIND command. The pipe character (|) tells the FIND command to look for *string* in the output generated by the CHKDSK command. The result of this command, then, is the full path name of every file on drive C that contains *string* somewhere within its name. Make sure you specify *string* in uppercase letters—otherwise the FIND command won't work.

For example, suppose you remember saving a letter to drive C under the file name GRISSOM.DOC, but you can't recall in which directory you've stored that file. Rather than hunt and peck through each directory on that drive, simply issue the command

```
    C>chkdsk c: /v | find "GRISSOM"
```

When you do this, you'll see the full path name of the file you're looking for. If DOS can't find a file whose name contains the string *GRISSOM*, the DOS prompt will reappear without displaying the file name.

Now, let's consider a similar situation in which this technique can really come in handy. Suppose you recall saving a worksheet file to your disk about a month ago, but you can't recall the worksheet's exact file name. All you know is that the file has a .WK1 extension, and it doesn't appear in the same directory as your other worksheet files. To locate the elusive file, issue the command *chkdsk c: /v | find "WK1"*. When you do this, DOS will show you the full path name of every file on your disk that has the extension .WK1.

```
    C>chkdsk c: /v | find "WK1"
            C:\MISC\BROWN.WK1
            C:\123\SALES.WK1
            C:\123\JUNK2.WK1
            C:\123\SHEET1.WK1
            C:\123\BUDGET90.WK1
            C:\123\PMTS.WK1
            C:\123\STUFF.WK1

    C>_
```

As you can see, most of the .WK1 files are stored in the directory *C:\123*. One file, however—BROWN.WK1—appears in the directory *C:\MISC*. When you notice this, you realize that BROWN.WK1 is, in fact, the file you are looking for.

Using your computer's system clock to your advantage

As you probably know, your computer has a system clock that keeps track of the current date and time. When the clock strikes midnight, the system clock advances to the next day, and if necessary, to the next month and year. Fortunately, the system clock is smart enough to remember how many days are in each month and which years are leap years.

Each time you create or modify a file, DOS pulls the current date and time from your computer's system clock, and "time stamps" that file. When you use the DIR command to see a directory listing of the current drive, you'll see the date and time when each file was last modified, as shown in Figure A. For instance, the first file listed in Figure A, AUTOEXEC.BAT, was last modified on 3-6-90 at 2:58 p.m.

Figure A

```
C>dir

 Volume in drive C has no label
 Directory of  C:\

AUTOEXEC BAT       290   3-06-90   2:59p
CONFIG   SYS       142   3-19-90   5:35p
COMMAND  COM     25308   7-18-88  12:00a
123          <DIR>       4-02-90  11:28a
WORD         <DIR>       4-28-90  11:29a
        5 File(s)  11315200 bytes free

C>_
```

DOS uses your system clock to "time stamp" files and directories.

As you can see in Figure A, DOS also associates a date and time with the directories on your drive. Of course, the date and time associated with a directory indicate when you originally created that directory.

The date and time information associated with your files and directories comes in handy in a variety of situations. For instance, if you want to delete all of the unnecessary data files on a particular disk, you can use the date and time information to determine quickly which files haven't been modified for awhile. Similarly, if you forget the name of a file you were working with earlier that day, you can scan the directory listing for the files that are labeled with the current date.

Setting the date and time on your system clock

Because DOS uses your computer's system clock to perform a variety of important functions, such as time stamping new files and directories, you'll want to make sure that you keep it up to date. Fortunately, the system clock in most computers, including nearly all 286- and 386-based machines, runs all the time. When you turn off the computer, a small battery inside the computer powers the system clock, keeping it up to date until you turn on the computer once again. (By the way, the battery in most PCs will last around five years, and is also used to keep the PC's configuration settings up to date.)

However, many older computers, including the IBM PC and PC-XT computers, are not equipped with a battery. If your computer falls into this category, shutting it off will reset its system clock to midnight, January 1, 1980. When you turn the computer back on, it should prompt you for the current date and time.

Even if your system clock is powered by a battery, you'll occasionally need to update your date and/or time. For instance, when daylight savings time begins or ends, you'll want to roll your system clock forward or backward an hour. Fortunately, DOS makes it easy to update the current date and time on your system.

To specify the current date, just enter the command date. Immediately, DOS will display the current date and will prompt you for the new date:

```
C>date
Current date is Sun  4-01-1990
Enter new date (mm-dd-yy): _
```

At this point, you can change the current date by typing a new date in *mm-dd-yy* form. For instance, to change the current date to May 26, 1990, type *05-26-90.* If you want to leave the current date as it is, you can simply press ↵ to return to the DOS prompt.

As you might guess, you can change the current time by issuing the command time. When you do this, DOS will prompt you for the current time:

```
C>time
Current time is  12:38:43.91
Enter new time: _
```

At this point, you can either type a new time and press ↵, or you can press ↵ to leave the current time as it is. You don't need to supply all the detail you see in the time prompt—you can just supply the hour and minutes, or just the hour. (The hour must appear in 24-hour format.) For example, to specify the time 9:00 a.m., you don't need to enter *9:00:00.00*—you can just enter *9:00* or even *9.* To specify 1:30 p.m., just enter *13:30.* To specify midnight, enter *0.*

Creating a custom prompt with the PROMPT command

Did you know that DOS lets you customize your prompt? You can use the PROMPT command to specify any prompt you want, including one that displays custom text, the current directory, or the current date and time.

The form of the PROMPT command is simply

```
prompt newPrompt
```

where *newPrompt* is the string of characters you want to appear as your new prompt. For example, suppose you want DOS to display the prompt *What is your wish?*, followed by a space. To do this, simply type the command *prompt What is your wish?* and press ↵ (make sure you type a space at the end of the command). When you do this, DOS displays the new prompt you've specified:

```
C>prompt What is your wish?

What is your wish? _
```

You can include any letter, number, or symbol you want in the *newPrompt* argument except the symbols

```
< > | $
```

If you want to use one of these symbols as part of your prompt, you must use a special code to do so. Let's take a look at all the special codes that are available to the PROMPT command.

Special prompt codes

DOS offers several special codes that you can include in the *newPrompt* argument. As you can see in Table A, all these codes consist of a dollar sign ($) and a single character. For instance, the code that tells DOS to display the current time is *$t*. Consequently, if you type the command *prompt The time is $t* and press ↵, DOS will display the current time as your prompt:

```
C>prompt The time is $t

The time is 10:31:08.10 _
```

As you can see, DOS includes much more precision in the time display than you'll probably want to see. If you want, you can reduce the amount of precision DOS displays by using the code *$h*. This code erases the character that immediately precedes that code in the prompt. For example, to reduce the time display to only hours and minutes, simply include six *$h* codes immediately after the *$t* code:

```
C>prompt The time is $t$h$h$h$h$h$h

The time is 10:36 _
```

As we mentioned a moment ago, you can include the symbols <, >, |, or $ in your prompt by including the codes that represent these symbols. For example, to display a prompt that includes the current drive and directory, followed by a greater than symbol (>), you can type the command *prompt $p$g* and press ↵:

```
C>prompt $p$g

C:\DOS>_
```

Although you might want to vary your prompt from time to time, you should always include the current drive and

Table A

This code...	**inserts...**
$p	the current drive and directory
$n	the current drive
$d	the current date (preceded by the day of week)
$t	the current time
$v	the DOS version number
$g	a greater than sign (>)
$l	a less than sign (<)
$b	a vertical bar (|)
$$	a dollar sign
$q	an equal sign (=)
$_ (underscore)	a newline character
$h	a backspace, which erases the previous character

You can display all kinds of useful information in your DOS prompt by including these special codes in the argument you supply to the PROMPT command.

directory information in your DOS prompt. That way, you can easily find your way around the subdirectories on your hard disk.

Two of the codes in Table A need a little explanation—the *$_* (newline) and *$h* (backspace) codes. When you include the code *$_* in your *newPrompt* argument, DOS will end the current line and will start a new one. For instance, the command *prompt line1$_line2$_line3* will create a three-line prompt:

```
C>prompt line1$_line2$_line3

line1
line2
line3_
```

You'll probably want to take advantage of the $_ code when you include several bits of information in your prompt. For example, to include the date, time, and current drive and directory in your prompt, you might precede each code representing these items with a *$_* code:

```
C>prompt $d$_$t$_$p$_

Tue  3-20-1990
10:33:00.48
C:\_
```

Resetting the prompt

If you do not specify a *newPrompt* argument in your PROMPT command, DOS will reset the prompt to its default—the current drive followed by a greater than sign. In other words, the two commands

```
C>prompt

C>prompt $n$g
```

produce the exact same result.

Conclusion

You can use DOS's PROMPT command to create just about any prompt you want. However, because the current drive and directory play essential parts in so many DOS commands, we suggest that you include this information in your custom prompt. To do this, simply issue the command:

```
C>prompt $p$g
```

This command tells DOS to display the current drive and directory, followed by a greater than sign (>).

In future issues of *Inside DOS*, we'll show you some advanced uses for the PROMPT command, such as how to redefine your function keys and how to customize the color of your screen.

Letter from the Editor

Welcome to *Inside DOS*, The Cobb Group's monthly newsletter designed to make you a more productive DOS user. As Editor-in-Chief, let me point out a couple of significant advantages to subscribing to our journal.

Nearly every month we will publish a couple of letters from our readers, along with our responses. Please write us! If you have a troubling DOS question, send it in—we might be able to help. If you want to comment on any of the articles we've published in a previous issue, please send us your feedback.

As you become more experienced with DOS, you'll undoubtedly discover some helpful timesaving tips or techniques. Don't be selfish and keep your discoveries secret! Share them with our readers. If you send us a tip or technique that we develop into a published article, we'll give you the byline and pay you between $25 and $100, depending on the complexity and usefulness of the tip. Keep in mind that you need not send in a polished article—simply jot down enough information to explain the tip or technique, and mail it in. Be sure to include your daytime phone number and mailing address with your submissions.

We hope you'll take the time to write us as often as you like. Send all correspondence to:

Editor-in-Chief, *Inside DOS*
The Cobb Group
P. O. Box 35160
Louisville, KY 40232-5160

Mark W. Crane

Mark W Crane

Editor-in-Chief

Using the PATH command to access external commands

One of the most important commands in DOS is the PATH command. If your computer has a hard disk that is organized into several directories, the PATH command makes it much easier to execute commands and to run applications. In this article, we'll explain the purpose of the PATH command and show you how it to use it to your advantage.

How DOS searches for commands

When you type something at the DOS prompt and press ↵, DOS assumes you've typed the name of a command. It then attempts to carry out the command by following these steps:

1. DOS checks to see if you typed the name of a built-in DOS command, like DIR or COPY. If you did, DOS will execute that command.
2. If you did not type the name of an internal command, DOS will check to see if you typed the name of a file stored in the current directory with the extension .COM or .EXE (a command file). If you did, DOS loads the command file into memory, and executes its instructions.
3. If you did not type the name of a command file, DOS will check to see if you typed the name of a file stored in the current directory with a .BAT extension (a batch file). If you did, DOS will execute the instructions in that batch file.
4. If you typed a command that does not match the name of a command file or a batch file stored in the current directory, DOS will display the message *Bad command or file name.*

Notice that DOS will look only in the *current directory* for command files and batch files. This means that in order to use an external DOS command, like FORMAT or CHKDSK, you'll need to make current the directory that contains your DOS command files. As you can imagine, this process is not practical.

The PATH command

Fortunately, DOS provides a command called PATH that makes it possible to access a command file or batch file that does not exist in the current directory. The PATH command defines a system path that tells DOS where to look for a command file or batch file if it cannot find that file in the current directory.

The form of the PATH command is

```
path  path1;path2;...;pathN
```

where *path1*, *path2*, and so forth are the full path names of the directories in which you want DOS to look. For example, the command

```
C>path c:\dos;c:\123;c:\batch
```

says to DOS, "If you can't find a command file or batch file in the current directory, look in the directory C:\DOS; if you can't find it there, look in the directory C:\123; finally, look for it in the directory C:\BATCH."

By way, if you want to examine your current path, just issue the command

```
C>path
```

without providing any arguments, and DOS will display the current path.

Which directories should you include in the path?

As we mentioned earlier, many DOS commands are external commands. To gain access to those commands regardless of your current drive or directory, you will need to include in your system path the name of the directory containing your DOS files.

Like DOS, applications often need to access command files while they are running. For this reason, you may want to include in your system path all the directories on your hard disk that contain application files. For instance, the path we defined a moment ago includes the directory C:\123, which is where we've installed Lotus 1-2-3 program files.

Once you've set your system path, DOS will remember that path until you reboot your computer. For this reason, it's a good idea to include a PATH command in your AUTOEXEC.BAT file, which is the file that DOS executes automatically upon rebooting. We'll discuss the AUTOEXEC.BAT file in detail in a future issue of *Inside DOS.*

Finally if you've created some batch files, you'll want to store those files in a single directory (C:\BATCH), and then include that directory in the system path, as we did in our example.

Conclusion

The PATH command allows you to access command files and batch files that are not stored in the current directory. If you use this command to include your DOS directory in the system path, you can successfully access external DOS commands and batch commands anytime you want. In a future issue of *Inside DOS,* we'll show you how to use the AUTOEXEC.BAT file to set your system path automatically each time you reboot.

DOS TIP

Save keystrokes using the [F3] key

Sometimes the simplest tips are also the most useful. The following tip offers a case in point.

Often when you're using DOS to manage the files on your hard disk, you'll need to issue a series of similar commands. For instance, if you're deleting a bunch of files in a directory one by one, you'll need to issue several ERASE or DEL commands in a row. On the other hand, if you type a long command and press ↵, then discover that you've accidentally mistyped a character, you'll need to retype the entire command. In situations like these, you can save yourself some keystrokes (and time) by using DOS's handy shortcut key—[F3].

Whenever you press the [F3] key while the cursor is at the DOS prompt, DOS will redisplay your previous DOS command. At that point, you can either press ↵ to reissue that command, or you can use the [Backspace] key to erase the characters one by one, and then type new characters. If you press [F3] again after editing the command, DOS will restore the portion of the command lying to the right of the cursor. To issue the corrected command, of course, just press ↵.

Let's demonstrate a situation in which the [F3] key is a real timesaver. Imagine you want to copy the file BUDGET91.WK1 (which is stored on the disk in your A: drive) to the directory C:\LOTUS\DATA. To do this, you'll need to issue the command:

```
C>copy a:budget91.wk1 c:\lotus\data
```

Instead of typing the proper command, however, let's suppose you accidentally type *lotud* instead of *lotus*:

```
C>copy a:budget91.wk1 c:\lotud\data
```

As soon as you press ↵, DOS will display the message *Invalid directory*. After inspecting your command line, you discover your typing error. Instead of retyping the entire command, you can first press [F3] to redisplay the mistyped command, then use the [Backspace] key to back up and erase the letter *d* in *lotud*. At this point, the command line will look like this:

```
C>copy a:budget91.wk1 c:\lotu
```

Now, type the letter *s*, and press [F3] again to redisplay the rest of the command. Finally, press ↵ to issue the corrected command and copy the file. ■

DOS TECHNIQUE

Redirecting DOS output from the screen to the printer

As you know, many DOS commands create reports. For instance, the DIR command lists the names of the files in the current directory, and the TREE command gives you a picture of your disk's directory structure. By default, DOS sends these reports to the standard output device—the screen. If you want, however, you can tell DOS to send a command's output to your printer instead of the screen. This technique is called *redirecting* a command's output.

It's easy to redirect a command's output to your printer. Simply follow the command with a greater than sign (>) and the device name PRN. For example, to print a list of files in the current directory, make sure your printer is powered on and is on line, then enter the command

```
C>dir > prn
```

As soon as you do this, DOS will print a directory listing like the one you're accustomed to seeing on the screen. (Notice that the redirecting options resemble an arrow pointing to the alternate device name PRN.) If you want to interrupt the printing after you've started it, simply press [Ctrl]c or [Ctrl][Break].

As we mentioned, the device name PRN represents your printer. This device name actually represents DOS's default parallel port—LPT1. If your printer is connected to port LPT1, the command *dir > prn* will have the same result as the command *dir > lpt1*. If your printer is connected to port LPT2 or LPT3, you'll want to use the device names LPT2 or LPT3 in your redirecting command. For example, to send a directory listing to the printer connected to port LPT3, simply issue the command

```
C>dir > lpt3
```

As you might expect, redirecting output has many more uses than just sending output to a printer. We will explore many other redirecting techniques in future issues of *Inside DOS*. ■

VAN WOLVERTON

Contributing Editor Van Wolverton is author of the bestselling books Running MS-DOS *and* Supercharging MS-DOS.

DOS: The Rodney Dangerfield of software

DOS is the Rodney Dangerfield of software: It may be the most widely used program in the world, but it gets no respect. What other program claims millions of users, yet receives 1000 complaints for every compliment? When was the last time you said to yourself "Boy, is DOS a good program!" Have you ever said it? Have you ever heard *anyone* say it?

Without DOS, application programs such as word processors and spreadsheets can't run, yet they get all the attention. If you're like the majority of DOS users, you work to become skilled—even expert—at using one or two programs. But far fewer users take the time to learn much about DOS. Some people, in fact, never even see DOS because the store where they bought their computer (or a helpful friend) installed both DOS and a menu system on their new machine so that it displays a menu each time it starts. To them, DOS is invisible—and forgettable.

Many others view DOS, at best, as a commute—something unproductive to be endured on the way to work. That's too bad, because skill in using DOS can make you more productive; sometimes it can even help you avoid losing time or valuable data.

There are several ways to learn more about DOS. Although DOS manuals usually aren't much help, there are some pretty good books and magazines available that are designed to make you a DOS expert. Alternatively, skilled users are an excellent source of help; fellow workers can often answer your questions, and a users' group is usually loaded with power users who know DOS inside out. Finally, classes are available through computer stores, adult education programs, and community colleges.

Unfortunately, these resources have significant disadvantages. Specifically, books go on for hundreds of pages, yet don't always answer your question; magazines contain mostly product reviews, opinion columns, and advertisements; skilled users may know the answer to your question, but often prefer just to "fix it" rather than take the time to explain, so that next time you can do it yourself; and classes are typically expensive.

Enter *Inside DOS*—The Cobb Group's monthly journal for DOS users. Every month, it shows you how to develop your DOS skills. Nothing else—just DOS. In my monthly column, and throughout this journal, you'll find articles on such topics as:

- techniques for managing your machine and files with DOS commands
- tips about little-known ways to bend DOS to your will
- batch files that can save you time by automating frequent or complicated command sequences

In this first column, I have two tips for you. The first one enables you to find your subdirectories quickly. The second explains alternative ways to display your file names.

Displaying subdirectories

Have you ever wanted to see the names of just the subdirectories in a directory? You can display all the directory entries and look for *<DIR>* in the output, of course, but there's an easier way. If you always include an extension when you name your files, but never when you name a directory, you can display the names of just the subdirectories by typing

```
C>dir *.
```

as we've done in Figure A.

Figure A

```
C>dir *.

 Volume in drive C: has no label
 Directory of  C:\

UTIL         <DIR>        8-11-89  10:27a
FASTBACK     <DIR>        8-11-89   1:11p
DOS          <DIR>        9-15-89   1:20p
EXCEL        <DIR>        9-21-89   1:21p
NORTON       <DIR>       10-08-89   9:27a
HPAD         <DIR>       11-02-89  12:05p
GAMES        <DIR>        2-13-90   5:10p
SNAP         <DIR>        2-26-90   8:30a
WORD5        <DIR>        3-10-90   1:23p
WINWORD      <DIR>        3-17-90   3:16p
TEMP         <DIR>        4-02-90  11:18a
MOUSE1       <DIR>        3-37-90  12:45p
BATCH        <DIR>        8-11-89   4:32p
QPRO         <DIR>        4-14-90   9:23a
PCLFONTS     <DIR>        8-11-90   4:45p
TD           <DIR>        8-11-89   4:23p
       16 File(s)   10950656 bytes free

C>_
```

You can display only the names of subdirectories by typing dir *. *at the DOS prompt.*

Figure B

```
C>dir /w
 Volume in drive C has no label
 Directory of  C:\

COMMAND  COM    VGA      COM    RAMBIOS  SYS    UTIL            FASTBACK
RAMDRIVE SYS    AUTOEXEC BAT    CONFIG   OLD    SMARTDRV SYS    TREEINFO NCD
SD       INI    COMM     DRV    DISPLAY  DRV    DOS             EXCEL
NORTON          HPAD            GAMES           SNAP            WORD5
WINWORD         HIMEM    SYS    TEMP            MOUSE1          NAMES    DOC
BATCH           QPRO            ASCII    DOC    AUTOEXEC OLD    PCLFONTS
TD              CONFIG   SYS
       32 File(s)  10946560 bytes free

C>_
```

To display a wide listing of your files and directories, issue the command dir /w.

You probably know that *.* means any file whose name includes any extension; what's lesser known is that *. means any file whose name does *not* include an extension. So, if your file names always include an extension and your directory names never do, typing *dir *.* displays only the names of directories.

Displaying wide and screen-by-screen directories

Keep in mind that DOS provides a couple of ways to manage those directories that swell with dozens of files. You can display up to 110 file names on one screen by using the /w (for *wide*) parameter with the DIR command; it displays just the file names and extension—none of the other information the DIR command usually displays—in five columns. To display the entries for the current directory in wide format, you'd type:

```
C>dir /w
```

as we've done in Figure B.

But what if you want to see other information, such as the size of the file or the date it was created or last changed? To do this, use the /p (for *pause*) parameter, which displays one screen full of directory entries, and then displays the message *Strike a key when ready*... at the bottom of the screen. To display the entries for the current directory one screen at a time, you would type:

```
C>dir /p
```

Figure C shows a screen like the one you'll see when you issue this command.

Take as long as you want to read the screen, then press any key to display the next screen of directory entries. If you want to stop the command before DOS displays the last screen, press [Ctrl]c (or [Ctrl][Break]).

There's much more to come. DOS is no more difficult to learn than most application programs—easier, in fact, than some. And you really aren't in charge of your system until you understand DOS and use it as well as you use your other programs. I'd appreciate your suggestions about this column. When possible, I'll try to address your questions. You can write me at The Cobb Group's address, which appears on page 2, or contact me via MCI Mail, or CompuServe (76655,236). ▩

Figure C

```
COMMAND  COM    25308   7-18-88  12:00a
VGA      COM     9057   8-11-89  10:26a
RAMBIOS  SYS    13997   8-11-89  10:26a
UTIL         <DIR>      8-11-89  10:27a
FASTBACK
RAMDRIVE SYS     8219   5-27-88  12:00a
AUTOEXEC BAT      290   3-06-90   2:58p
CONFIG   OLD      119   1-31-90   1:48p
SMARTDRV SYS    10209   8-11-89   3:47p
TREEINFO NCD      235   2-21-89   8:13p
SD       INI     2497   2-21-89   8:22p
COMM     DRV     4676  12-14-88  12:19p
DISPLAY  DRV    44272   2-07-89  10:51p
DOS          <DIR>      9-15-89   1:20p
EXCEL        <DIR>      9-21-89   1:21p
NORTON       <DIR>     10-08-89   9:27a
HPAD         <DIR>     11-02-89  12:05p
GAMES        <DIR>      2-13-90   5:10p
SNAP         <DIR>      2-26-90   8:30a
WORD5        <DIR>      3-10-90   1:23p
WINWORD      <DIR>      3-17-90   3:16p
HIMEM    SYS
TEMP         <DIR>      4-02-90  11:18a
Strike a key when ready . . .
```

To display your directory entries one screen at a time, type dir /p.

CHRIS DEVONEY

An introduction to batch files

The following article was written by Contributing Editor Chris DeVoney. Chris is the author of several DOS books, including DOS Tips, Tricks, and Traps, *and is a contributing editor to* PC/Computing *magazine.*

Do you repeatedly issue the same set of commands to start your favorite programs? Are you tired of typing a particularly long or hard-to-remember command? Do you often misspell commands? If so, you should take advantage of one of DOS's most powerful facilities—batch files. By learning how to create and execute batch files, you can save time, tedium, and aggravation.

What is a batch file?

A batch file is a text file with a .BAT file-name extension that contains a series of commands—commands you'd normally type at the DOS prompt. The file can contain just one command, or hundreds of commands. When you tell DOS to execute a batch file, it will execute all the commands in that file one by one, just as if you typed those commands from the keyboard.

To execute a batch file, you type its name at the DOS prompt (without the extension .BAT) and press ↵. For instance, if you store a series of commands in a batch file named LIST.BAT, you can automatically execute all the commands in that file by typing the command *list* at the DOS prompt and pressing ↵.

An example

Let's create and execute a simple batch file. Suppose you want to create a batch file that activates the directory C:\DOS, and that lists all the files in that directory that begin with the letter *F.* To do this from the DOS prompt, of course, you first issue the command *cd \dos* to move into the directory C:\DOS, then issue the command *dir f** to list the files whose names begin with *F.*

Figure A

```
C>copy con list.bat
cd \dos
dir f*
^Z
        1 File(s) copied

C>_
```

You can use the COPY CON command to create simple batch files.

To create a batch file that performs these two tasks for you, type the commands shown in Figure A (the ^Z characters represent the key combination [Ctrl]z):

The command *copy con list.bat* tells DOS to store the commands you type in a file named LIST.BAT. Of course, the lines *cd \dos* and *dir f** are the commands you want to store in the batch file. To tell DOS that you're finished storing commands in LIST.BAT and that you want to return to the DOS prompt, you simply press [Ctrl]z (which displays the characters ^Z), then press ↵.

Now, to run the batch file you've created, simply issue the command

```
C>list
```

Immediately, DOS will issue the commands *cd \dos* and *dir f**, as shown in Figure B.

Figure B

```
C>list

C>cd \dos

C>dir f*

 Volume in drive C has no label
 Directory of  C:\DOS

FASTOPEN EXE     3919   3-17-87  12:00p
FDISK    COM    48216   3-18-87  12:00p
FIND     EXE     6434   3-17-87  12:00p
FORMAT   COM    11616   3-18-87  12:00p
        4 File(s)  10629120 bytes free

C>
C>_
```

When you run the batch file you created in Figure A, DOS will execute both the cd \dos *and the* dir f* *commands.*

As you can see, this batch file produces a double DOS prompt when it finishes running. If you want to eliminate the double prompt, simply use the [Ctrl]z combination to enter the characters ^Z at the end of the last line in the batch file instead of entering them on a line by themselves.

Rules of the game

Now that you have a better feel for batch files, let me point out a few rules you will want to keep in mind as you create your own batch files.

1. Use a .BAT file-name extension

DOS demands that a batch file have a file-name extension (suffix) of .BAT. You can assign a batch file any root name that DOS will accept, but failing to supply a .BAT suffix will prevent DOS from being able to execute that file.

2. Choose names that don't conflict with other command names

Make sure you carefully choose the names for your batch files. You must not assign a root name to a batch file that matches either a built-in DOS command (like DIR or COPY) or an existing command file (a file with a .COM or .EXE file-name extension). If you violate this rule and create a conflict between a batch file name and another command name, DOS will not be able to execute that batch file.

3. Use the COPY CON command to create short batch files

If you want to create a batch file that is only a few lines long, you should create it using the COPY CON command, which we demonstrated a moment ago. The form of this command is

```
copy con filename.bat
```

where *filename* is the root name of the batch file you want to create. When you enter this command, the cursor will drop to the next line. At this point, you should type the first line for the batch file, press ↵, type the second line, and so forth until you've entered every command you want to store.

Watch your typing! When you use the COPY CON command, you can only edit the current line as you go along. If you press ↵ to advance to the next line, then notice a mistake on a previous line, you'll need to quit and start over again (or complete the file, load a text editor, and edit the mistake from the file).

After you've entered all your lines, press [Ctrl]z. (If you want, you can press the [F6] key, which enters the same character as the [Ctrl]z key combination.) When you do this, the characters *^Z* will appear on the screen. Next, press ↵ again. At this point, DOS will save the batch file to disk and will display the message *1 file(s) copied.* Now, the batch file is ready to run.

4. Be careful with word processors

As we mentioned, a batch file contains only text that you can enter at the DOS prompt. When you want to edit the text in a batch file, you can use either a simple text processor or a full-featured word processor. Fortunately, DOS provides a simple text editor called EDLIN, which is perfect for creating and editing batch files. We'll discuss EDLIN in future issues of *Inside DOS.*

If you use a word processor to create and edit batch files, be careful. Most word processors insert special formatting codes into their documents when they save them. When DOS attempts to execute a batch file containing extra formatting codes, problems will inevitably occur.

To make sure your word processor does not include formatting codes in its documents, you must save them as ASCII text. You will need to refer to your word processor manual to find out how to do this. Some word processors refer to the ASCII format as the "PC-8", or "non-document format."

5. Make sure your batch files are on the system path

You'll want to make sure you can access your batch files as you move around in the subdirectories on your hard disk. For this reason, you should save all of your batch files in a single directory (like C:\BATCH or C:\UTIL), and then include that directory in your system path. The article entitled "Using the PATH Command to Access External Commands" on page 6 of this issue shows you how to include a directory in your path.

A tip: fixing misspelled commands

As I hinted earlier, batch files are ideal for helping you deal with commands that you frequently misspell. By creating a batch file whose root name matches the misspelled command, you can force DOS to do what you want—not what you type.

For instance, suppose you frequently type the directory command as *DRI* instead of *DIR.* To make DOS execute the DIR command even when you make this mistake, create a batch file called DRI.BAT that contains the command

```
dir %1 %2 %3 %4
```

(The arguments *%1* through *%4* tell DOS to recognize the arguments you supply to your misspelled DIR command.) Now, the next time you accidentally type

```
C>dri
```

DOS will still execute the *DIR* command.

There's much more

The concepts I've discussed in this article barely scratch the surface of batch files. As you'll see in future issues of this journal, batch files are very powerful tools that can make DOS much easier to use. If you find the concept of batch files a bit elusive, don't worry. As you write your own batch files and experiment with new features, you'll eventually find batch files as useful as they are powerful. ▨

Postmaster:
Monthly Technical Journal
Please Rush!

Please include account number from label with any correspondence.

DOS TECHNIQUE

Displaying the contents of a text file

DOS provides a couple of tools that make it easy to view the contents of any text file. You'll want to use these tools whenever you need to view the contents of either a batch file you've created or a README file that is included with an application program. Let's briefly look at these two tools: the TYPE and MORE commands.

The TYPE command

The TYPE command tells DOS to display the entire contents of the file you specify. The form of this command is

```
type filename
```

where *filename* is the name of the file whose contents you want to see. For example, if you want to view the contents of the batch file LIST.BAT, which we created on page 10, simply issue the command

```
C>type list.bat
```

Immediately, DOS will display the contents of that file, as shown in Figure A.

Figure A

```
C>type list.bat
cd \dos
dir f*

C>_
```

The TYPE command tells DOS to display the entire contents of a file.

If the contents of the file you're displaying are longer than a single page, DOS will scroll the first part of the file off the top of the screen before you have a chance to read it. In fact, the only part of the file you'll be able to read are the last 23 lines. Fortunately, you can overcome this problem with the help of the MORE command.

The MORE command

The MORE command lets you view the result of the TYPE command one screen at a time. To take advantage of the MORE command, simply append the pipe character (|) and the command toward the end of the TYPE command, like this:

```
type filename | more
```

When you issue this command, DOS will display the first 24 lines of the file you specify, and then will display the message

```
— More —
```

at the bottom of the screen to indicate that there is more information to display.

When you are ready to view the next screen, press any key. DOS will then display a blank line along with the next 23 lines of the file, and will once again display the message—*More*—at the bottom of the screen. DOS will display the entire file one screen at a time.

By the way, TYPE isn't the only command with which you can use the MORE command. You can pipe the output of any report-generating command to the MORE command. For example, as we explained in our lead article, the command

```
C>chkdsk c: /v
```

tells DOS to display the full path name of every file stored on drive C. If drive C contains more than one screen of file names, you can tell DOS to display them one screen at a time by piping the CHKDSK command's output to the MORE command, like this:

```
C>chkdsk c: /v | more
```

Cancelling the command

If you've used the MORE command to display a few screens of text, and you suddenly want to abort the command and return to the DOS prompt, simply press [Ctrl]c or [Ctrl][Break]. When you do this, DOS will immediately return you to the DOS prompt.